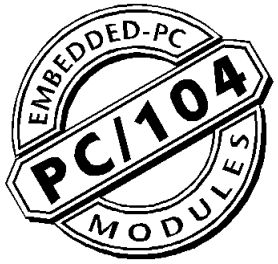


DM5854HR/DM6854HR
Isolated digital I/O-module
User's Manual

Real Time Devices Finland Oy

Real Time Devices Finland Oy
Lepolantie 14
FIN-00660 Helsinki, Finland
Tel: (+358) 9 346 4538
Fax: (+358) 9 346 4539
Email: sales@rtdfinland.fi
URL: www.rtdfinland.fi

(c) RTD Finland Oy 1996-2001
ISO9001 and AS9100 Certified

IMPORTANT

Although the information contained herein has been carefully verified RTD Finland Oy assumes no responsibility for any errors that may occur, for any damage to property or persons resulting from improper use of this manual or from the related software. RTD Finland Oy also reserves the right to alter the contents of this manual, as well as features and specifications of this product at any time, without prior notice.

Published by
Real Time Devices Finland Oy
Lepolantie 14
FIN-00660 Helsinki, Finland

Copyright © 1996-2001 by RTD Finland Oy
All rights reserved

Printed in Finland

Table of Contents

INTRODUCTION
- **General purpose digital I/O**
- **Isolated digital inputs**
- **Isolated digital outputs**
- **Mechanical description**
- **Connector description**
- **What comes with your board**
- **Board accessories**
  - **Application software and drivers**
  - **Hardware accessories**
- **Using this manual**
- **When you need help**

CHAPTER 1 - BOARD SETTINGS
- **Factory-configured jumper settings**
- **Base Address jumpers**
- **Interrupts and interrupt channels**
- **Pull-up or Pulldown jumpers**
- **Isolated input range selection jumpers**

CHAPTER 2 - BOARD INSTALLATION
- **Board installation**
- **External I/O connections**
  - **Isolated Input/Output Connector**
  - **50-pin RTD Expansion Connector**

CHAPTER 3 - HARDWARE DESCRIPTION
- **Digital I/O, 8255 Programmable digital I/O**
- **Isolated optocoupler inputs**
- **Isolated optocoupler outputs**
- **Trigger output**
- **Interrupts**

CHAPTER 4 - BOARD OPERATION AND PROGRAMMING

Defining the I/O map

Base+0	**Read digital inputs / Clear interrupt**
Base+1	**Digital input mask**
Base+2	**Digital input inversion register**
Base+3	**Control register / Digital outputs**
Base+4	**PPI Port A (Read/Write)**
Base+5	**PPI Port B (Read/Write)**
Base+6	**PPI Port C (Read/Write low and high byte)**
Base+7	**PPI Port CW (Write)**

Programming the DM6854HR

Clearing and setting bits in a port

Initializing the 8255 PPI

Isolated Digital Input programming

Isolated Digital Output programming

Trigger Output programming

Interrupts

- **What is an interrupt?**
- **Interrupt request lines**
- **8259 Programmable interrupt controller**
- **Interrupt mask register (IMR)**
- **End-Of-Interrupt (EOI) Command**
- **What exactly happens when an interrupt occurs?**
- **Using interrupts in your program**
- **Writing an interrupt service routine (ISR)**
- **Saving the startup IMR and interrupt vector**
- **Common Interrupt mistakes**

APPENDIX A - DM5854HR/6854HR Specifications

List of Illustrations

This user's manual describes the operation of the DM5854HR/DM6854HR Isolated Digital Interface board.

Some of the key properties of the DM5854HR/DM685HR include:

* 8 channel-by-channel isolated digital inputs with masking modes
* 4 Optocoupled digital outputs with open collector configuration, 30V range
* 24 non isolated TTL/CMOS compatible 8255-based digital I/O lines
* Optional Pull-up or Pull-down resistors
* Software enabled interrupts XT and AT interrupts available
* Single +5V operation
* Support for direct PC/104 interface with RTD dataModules®
* XT (DM5854HR) and AT (DM6854HR) boards available
* PC/104 compliant
* RTD IDAN compatible

The following paragraphs briefly describe the major features of the DM6854HR. A more detailed discussion is included in Chapter 3 (Hardware description) and in Chapter 4 (Board operation and programming. The board setup is described in Chapter 1 (Board Settings).

General purpose digital I/O

The DM6854HR board has 24 8255-based TTL/CMOS compatible general purpose digital I/O lines which can be directly interfaced with external devices or signals to interface to alphanumerical displays, sense switch closures, trigger digital events, or activate solid-state relays. These lines may be pulled down or pulled up or pulled down with 10K Ohm resistor networks. Installation procedures will be described later in this manual.

Isolated digital inputs

8 Optocoupler inputs may be used to connect high voltage signals to a computer. Eight channel-by-channel jumper configurable input ranges are available: +5V, +12V and +24V. The optocoupler inputs have a reverse voltage protection diode across the input. This enables AC-signals to be connected to the isolated optocoupler inputs. This input diode also acts as a rectifier. The programmable functions of the DM6854HR enable input masking and input pattern recognition with interrupts. This relieves the host computer from processor capacity consuming input polling.

Isolated digital outputs

4 open collector optocoupler outputs may be used to directly drive loads such as switches or relays with voltages up to 30V DC.

Mechanical description

The DM6854HR is designed on a PC/104 form factor. An easy mechanical interface to both PC/104 and EUROCARD systems can be achieved. Stack your PC/104 compatible computer directly on the DM6854HR using the onboard mounting holes.

Connector description

There is a 50 pin digital interface connectors on the DM6854HR to directly interface to the non-isolated 8255-based digital I/O signals. The signal definition of this connector is compatible with the digital PC/104 dataModules manufactured by RTD.

Isolated outputs and inputs are connected to the DM6854HR by either a screw terminal discrete wire connector of with a 50-pin flat ribbon cable header connector. Use this type of interface connector with a TB50 screw terminal block. Please consult the factory for more details on different connector options.

What comes with your board

You receive the following items in your DM6854HR package:

* DM5854HR or DM6854HR Isolated digital interface module
* User's manual

Note: Latest software and drivers can be downloaded from our website.

If any item is missing or damaged, please call Real Time Devices Finland Customer service department at (+358) 9 346 4538.

Board accessories

In addition to the items included in your DM6854HR delivery several software and hardware accessories are available. Call your distributor for more information on these accessories and for help in choosing the best items to support your control system.

* Application software and drivers

* Hardware accessories such as IDAN aluminium enclosure systems.
  For more information in IDAn please visit our websites at
  www.rtdfinland.fi and **www.rtdusa.com**.

Real Time Devices can supply a complete set of accessories to your DM6854HR card. These include enclosures, power supplies, terminal boards (TB50) and other interconnection systems.

Using this manual

This manual is intended to help you install your new DM6854HR card and get it running quickly, while also providing enough detail about the board and it's functions so that you can enjoy maximum use of it's features even in the most demanding applications.

When you need help

This manual and all the example programs will provide you with enough information to fully utilize all the features on this board. If you have any problems installing or using this board, contact our Technical Support Department (+358) 9 346 4538 during European business hours, or send a FAX to (+358) 9 346 4539 or Email to sales@rtdfinland.fi. When sending a FAX or Email request, please include your company's name and address, your name, your telephone number, and a brief description of the problem.

CHAPTER 1 - BOARD SETTINGS

The DM6854HR Isolated digital I/O board has jumper settings you can change to suit your application and I/O configuration setup. The board is factory configured with a +24V input range configuration. The factory settings are listed and shown in the diagram in the beginning of this chapter.

Factory-Configured Jumper Settings

Table 1-1 illustrates the factory jumper setting for the DM6864HR. Figure 1-1 shows the board layout and the locations of the jumpers. The following paragraphs explain how to change the factory jumper settings to suit your specific application.

Table 1-1 - Factory jumper settings see figure 1-1 for detailed locations

JUMPER NAME	DESCRIPTION OF JUMPER	NUMBER OF JUMPERS	FACTORY SETTING JUMPERS INSTALLED
BASE	BASE ADDRESS	5	300h INSTALLED
IRQ	INTERRUPT CHANNEL	6	5
PU/PD IN1-8	PU/PD FOR PPI PORTS INPUT RANGE FOR ISOLATED INPUTS 1-8	3-Terminal 3	PULLDOWN +24V RANGE

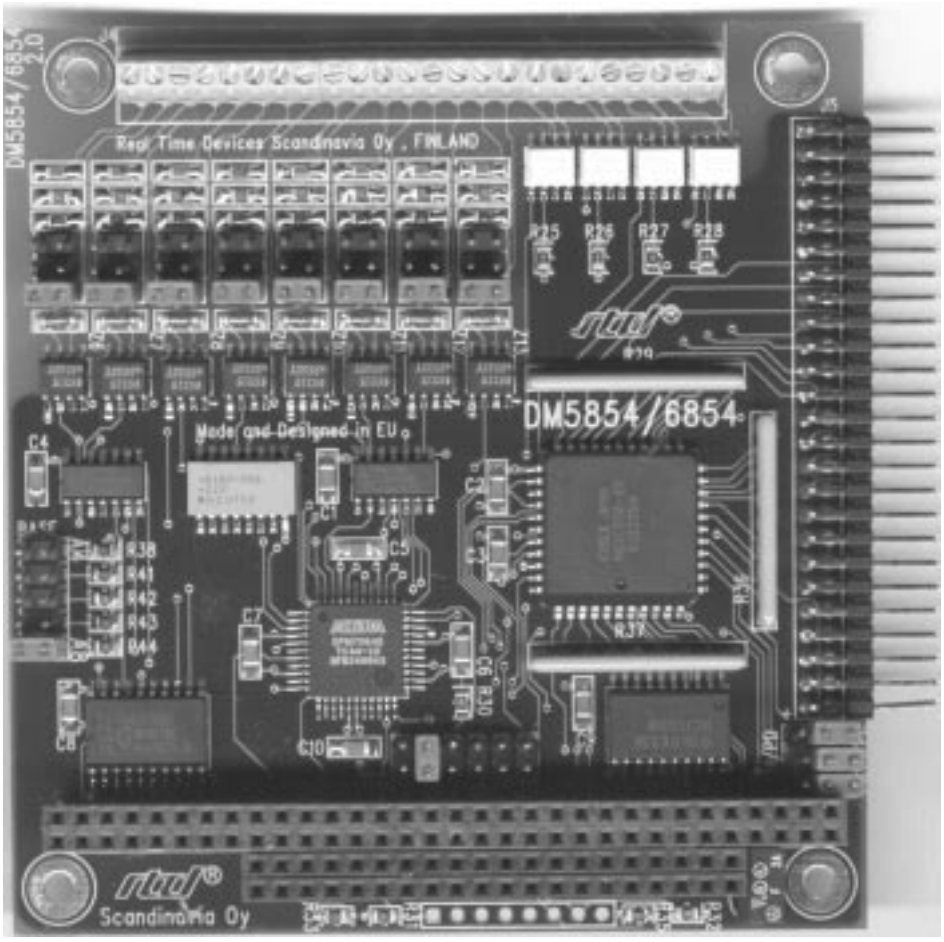


Figure 1-1 - Board layout showing jumper locations

Base Address jumpers (Factory setting: 300h)

The most common cause of failure when you are first setting up your module is address contention. Some of your computers I/O space is already occupied by other internal I/O devices and expansion boards. When the DM6854HR attempts to use it's reserved I/O addresses already used by another peripheral device erratic performance may occur and data read from the board may be corrupted.

To avoid this problem make sure you set up the base address first using the jumpers marked "BASE" which let you choose from 32 different I/O addresses in your computers I/O map. Should the factory installed setting of 300H be unusable for your system configuration, you may change this setting to another using the options illustrated in Table 1-2. The table shows the switch settings and their corresponding values in hexadecimal values. Make sure you verify the correct location of the base address jumpers. When you set the base address of the module, record the setting in the table inside the back cover of this manual after the Appendices.

Table 1-2 Base Address jumper settings

Base Address jumper settings DM6854HR			
	Jumper Settings A8 A7 A6 A5 A4	Base Address Hex / (Decimal)	Jumper Settings A8 A7 A6 A5 A4
200 / (512)	0 0 0 0 0	300 / (768)	1 0 0 0 0
210 / (528)	0 0 0 0 1	310 / (784)	1 0 0 0 1
220 / (544)	0 0 0 1 0	320 / (800)	1 0 0 1 0
230 / (560)	0 0 0 1 1	330 / (816)	1 0 0 1 1
240 / (576)	0 0 1 0 0	340 / (832)	1 0 1 0 0
250 / (592)	0 0 1 0 1	350 / (848)	1 0 1 0 1
260 / (608)	0 0 1 1 0	360 / (864)	1 0 1 1 0
270 / (624)	0 0 1 1 1	370 / (880)	1 0 1 1 1
280 / (640)	0 1 0 0 0	380 / (896)	1 1 0 0 0
290 / (656)	0 1 0 0 1	390 / (912)	1 1 0 0 1
2A0 / (672)	0 1 0 1 0	3A0 / (928)	1 1 0 1 0
2B0 / (688)	0 1 0 1 1	3B0 / (944)	1 1 0 1 1
2C0 / (704)	0 1 1 0 0	3C0 / (960)	1 1 1 0 0
2D0 / (720)	0 1 1 0 1	3D0 / (976)	1 1 1 0 1
2E0 / (736)	0 1 1 1 0	3E0 / (992)	1 1 1 1 0
2F0 / (752)	0 1 1 1 1	3F0 / (1008)	1 1 1 1 1

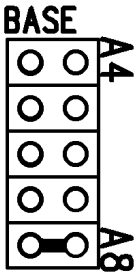


Fig. 1-2 Base Address jumper block (A4 to top, A8 to bottom; open <-> 0)

Interrupt channel (Factory setting: IRQ5)

The header connector, shown on Figure 1-3, lets you connect the board interrupt to one of the 6 interrupt channels available on the XT/AT-bus.

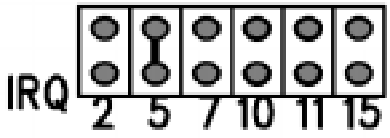


Fig. 1-3 Interrupt selection jumpers

The interrupt is caused by an external trigger event that has transferred the storage latch data to the output latch of the isolated digital output stage.

Pull-up or Pull-down resistor selection for digital I/O PA, PB, PC
(Factory setting: PA, PB and PC pulled down)

The 8255 programmable digital I/O interface provides 24 TTL/CMOS compatible lines which can be interfaced with external devices. These lines are divided into three groups: eight Port A lines, eight Port B lines and eight Port C lines. You can connect pull-up or pull-down resistor networks for ports A, B and C. You may want to pull lines up for connection with switches. This will pull the lines high if the switch is disconnected. Or you may want to pull lines down for connection to relays which control turning motors on or off. The port A, B and C lines of the 8255 programmable digital I/O interface are set as inputs after reset. This can cause the external devices connected to these lines to operate erratically. Pulling these lines down, when the board is powered up the lines will not be active before the 8255 is initialized. Figure 1-4 shows both Ports A, B and C set in the Pull-down configuration.

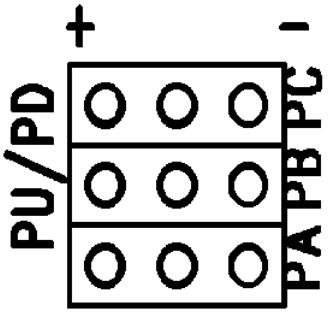


Fig. 1-4 Pull-up and Pull-down jumpers for Ports A, B and C.
(Inputs pulled down when jumpers in right-hand position, see above)

PC4-PC7 Optocoupler isolated inputs (Factory setting: +24V range)

The Optocoupler inputs can be jumper configured for different input ranges. Each channel can be set for a different range. The different ranges use different current limiting resistors at the optocoupler diode input. Make sure you set the range correctly since overloading the optocoupler with overcurrent may cause permanent damage to the device. All the channels are set in a similar manner. Figure 1-5 illustrates one channel set for a 24V-level input signal. Figure 1-6 illustrates the complete input stage of the DM6854HR. The jumper topmost toward the spring loaded connectors selects +5V input range, the middle position +12V and the bottom most (toward the center of the board) +24V. The optocoupler forward current is 10mA for switching (max. 16mA).

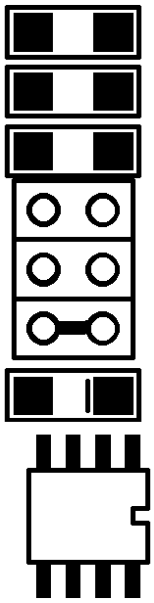


Fig. 1-5 Optocoupler input range jumpers set for 24V input range

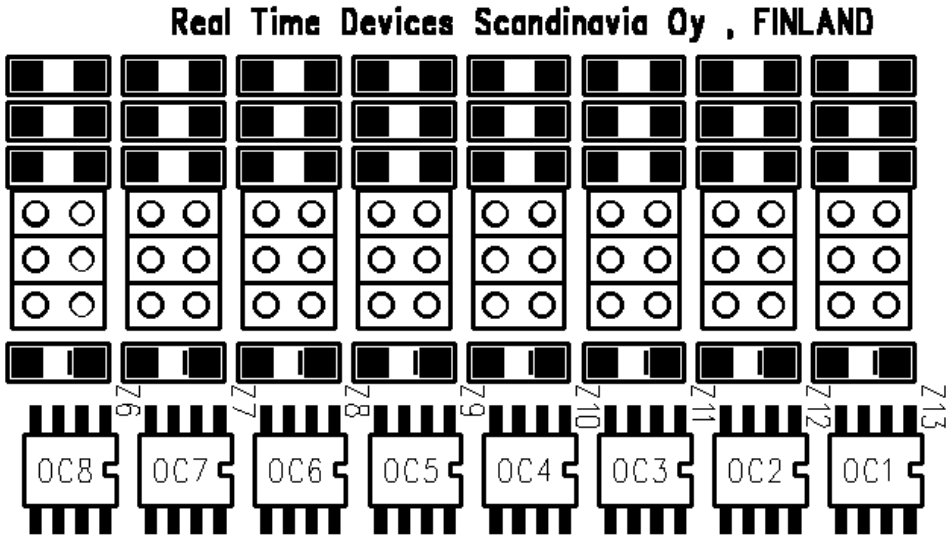


Fig. 1-6 Optocoupler input stage layout of the DM6854HR

CHAPTER 2 - BOARD INSTALLATION

The DM6854HR Isolated digital interface board is very easy to connect to your control system. Direct interface to PC/104 systems as well as EUROCARD boards is provided. This chapter tells you step-by-step how to install your board into your system.

After completing the installation use the diagnostic software to fully verify that your board is working.

Board Installation

Keep your board in its antistatic bag until you are ready to install it to your system! When removing it from the bag, hold the board at the edges and do not touch the components or connectors. Please handle the board in an antistatic environment and use a **grounded** workbench for testing and handling of your hardware.

Before installing the board in your computer, check the jumper settings. Chapter 1 reviews the factory settings and how to change them. If you need to change any settings, refer to the appropriate instructions in Chapter 1. Note that incompatible jumper settings can result in unpredictable board operation and erratic response.

General installation guidelines:

1. Turn OFF the power to your computer
2. Touch the grounded metal housing of your computer to discharge any antistatic buildup and then remove the board from its antistatic bag.
3. Hold the board by it's edges and install it in an enclosure or place it on the table on an antistatic surface.
4. Connect the board to the data acquisition board using the twisted pain 50-pin flat cable. Make sure that the polarity of the cable is correct.

Installation integrated with a PC/104 module stack:

* Secure the four PC/104 installation holes with standoffs.

* Connect the 50-pin expansion connector to your nonisolated digital signals and directly connect your isolated input and output control devices to the onboard I/Oconnectors.

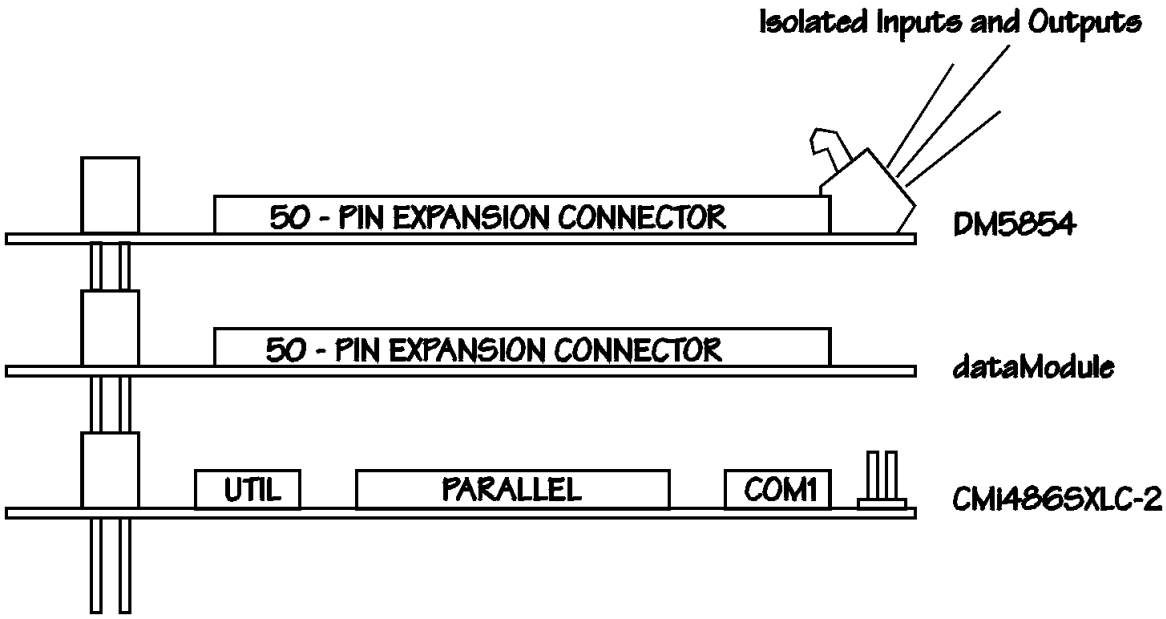


Fig. 2-1 DM5854HR integrated with a PC/104 dataModule® stack.

3U rack or enclosure installation with a EUROCARD CPU with one DM5854HR.

* The PC/104 system can easily be inserted into a 19" rack installation using the CPU as a "form factor adapter". Assemble your PC/104 dataModules® on a RTD single board EUROCARD computer and install the system in a 19" enclosure. Multiple DM5854HR boards can easily be connected to this system. See figure 2-2.

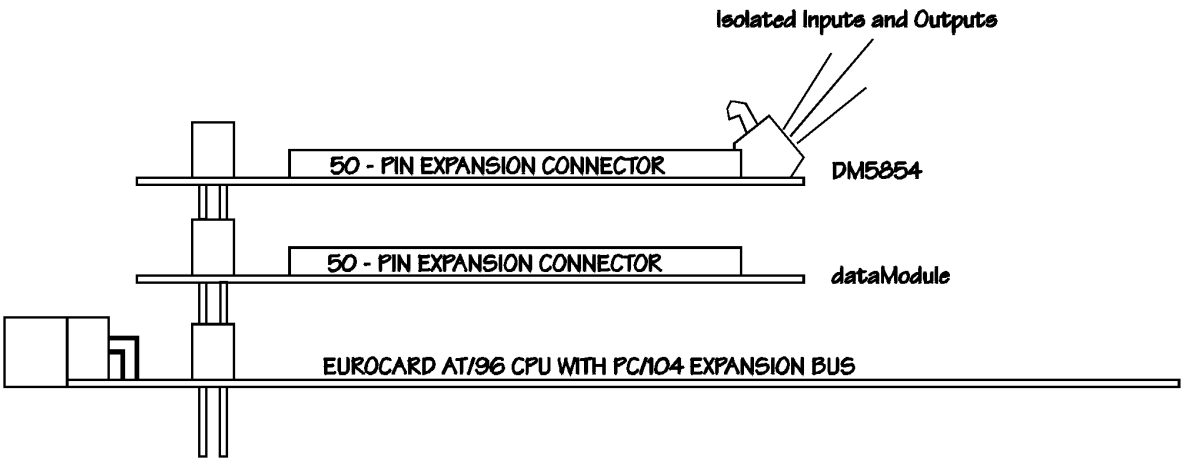


Fig. 2-2 19" Eurocard rack installation with an integrated PC/104 dataModule and EUROCARD cpuModule computer system.

External I/O connections

Figure 2-3 shows the interface connector layout of the DM6854HR. This connector is located toward the top of the board. Refer to this diagram when making isolated signal output connections. Figure 2-4 shows the standard I/O connector pinout.

Figure 2-3a shows the screw terminal discrete wire connector pinout and figure 2-3b shows the pinout of the same connector using a header connector.

Pin 1 is to the left side of the connector. ALL SIGNALS ARE ISOLATED

1. IN1+
2. IN1-
3. IN2+
4. IN2-
5. IN3+
6. IN3-
7. IN4+
8. IN4-
9. IN5+
10. IN5-
11. IN6+
12. IN6-
13. IN7+
14. IN7-
15. IN8+
16. IN8-
17. OUT1_COLLECTOR
18. OUT1_EMITTER
19. OUT2_COLLECTOR
20. OUT2_EMITTER
21. OUT3_COLLECTOR
22. OUT3_EMITTER
23. OUT4_COLLECTOR
24. OUT4_EMITTER

Fig. 2-3a Isolated Input/Output interface using screw terminal block connector

#1,2	IN1+
#3,4	IN1-
#5,6	IN2+
#7,8	IN2-
#9,10	IN3+
#11,12	IN3-
#13,34	IN4+
#15,16	IN4-
#17,18	IN5+
#19,20	IN5-
#21,22	IN6+
#23,24	IN6-
#25,26	IN7+
#27,28	IN7-
#29,30	IN8+
#31,32	IN8-
#33,34	OUT1_COLLECTOR
#35,36	OUT1_EMITTER
#37,38	OUT2_COLLECTOR
#39,40	OUT2_EMITTER
#41,42	OUT3_COLLECTOR
#43,44	OUT3_EMITTER
#45,46	OUT4_COLLECTOR
#47,48	OUT4_EMITTER

Fig. 2-3b Isolated Input/Output interface using a 48 - pin header connector

Figure 2-4 shows the 50-pin expansion interface connector layout of the DM6854HR. This connector is used to interface to the non-isolated 8255 based digital I/O signals.

Pin 1 is away from the PC/104 connector.

PC0	*1	*2	EXT_TRIGGER (OUTPUT)
PC1	*3	*4	DGND
PC2	*5	*6	DGND
PC3	*7	*8	DGND
PC4	*9	*10	DGND
PC5	*11	*12	DGND
PC6	*13	*14	DGND
PC7	*15	*16	DGND
PB0	*17	*18	DGND
PB1	*19	*20	DGND
PB2	*21	*22	DGND
PB3	*23	*24	DGND
PB4	*25	*26	DGND
PB5	*27	*28	DGND
PB6	*29	*30	DGND
PB7	*31	*32	DGND
PA0	*33	*34	DGND
PA1	*35	*36	DGND
PA2	*37	*38	DGND
PA3	*39	*40	DGND
PA4	*41	*42	DGND
PA5	*43	*44	DGND
PA6	*45	*46	DGND
PA7	*47	*48	DGND
+5V (fused)	*49	*50	DGND

Fig. 2-4 50-pin expansion interface connector pinout

CHAPTER 3 - HARDWARE DESCRIPTION

Chapter 3 - **Hardware Description** describes the major features of the DM6854HR: the 8255 based digital I/O, isolated optocoupler inputs, isolated optocoupler outputs, interrupts and output triggering.

Figure 3-1 shows the general block diagram of the DM6854HR. This chapter describes the major features of the DM6854HR: the 8255 based digital I/O, isolated optocoupler inputs, isolated optocoupler outputs, interrupts and output triggering.

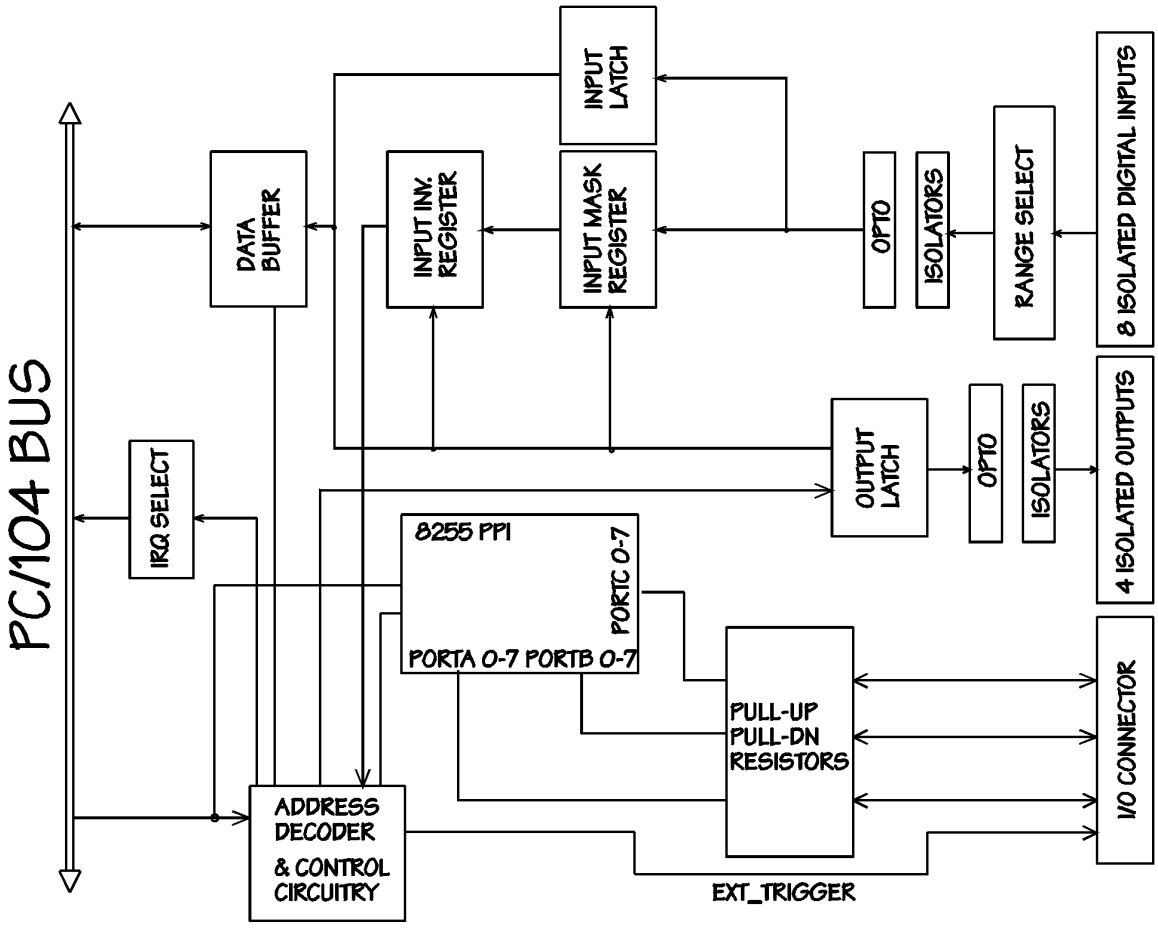


Fig. 3-1 DM6854HR Block diagram

Digital I/O, Programmable Peripheral Interface (PPI)

The programmable peripheral interface (PPI) is used for general purpose digital I/O functions. This high performance TTL/CMOS compatible chip has 24 digital I/O lines divided into two groups of 12 lines each.

Group A: Port A (8) lines and Port C upper (4) lines
Group B: Port B (8) lines and Port C lower (4) lines

Port A, Port B and Port C are available at the 50 pin expansion connector. You can use Ports A , B and C in one of these three operating modes:

Mode 0: Basic I/O. Lets you use simple input output functions for a port. Data is written to or read from the specified port

Mode 1: Strobed Input/Output. Lets you transfer data I/O from port A in conjunction with strobed or handshake signals.

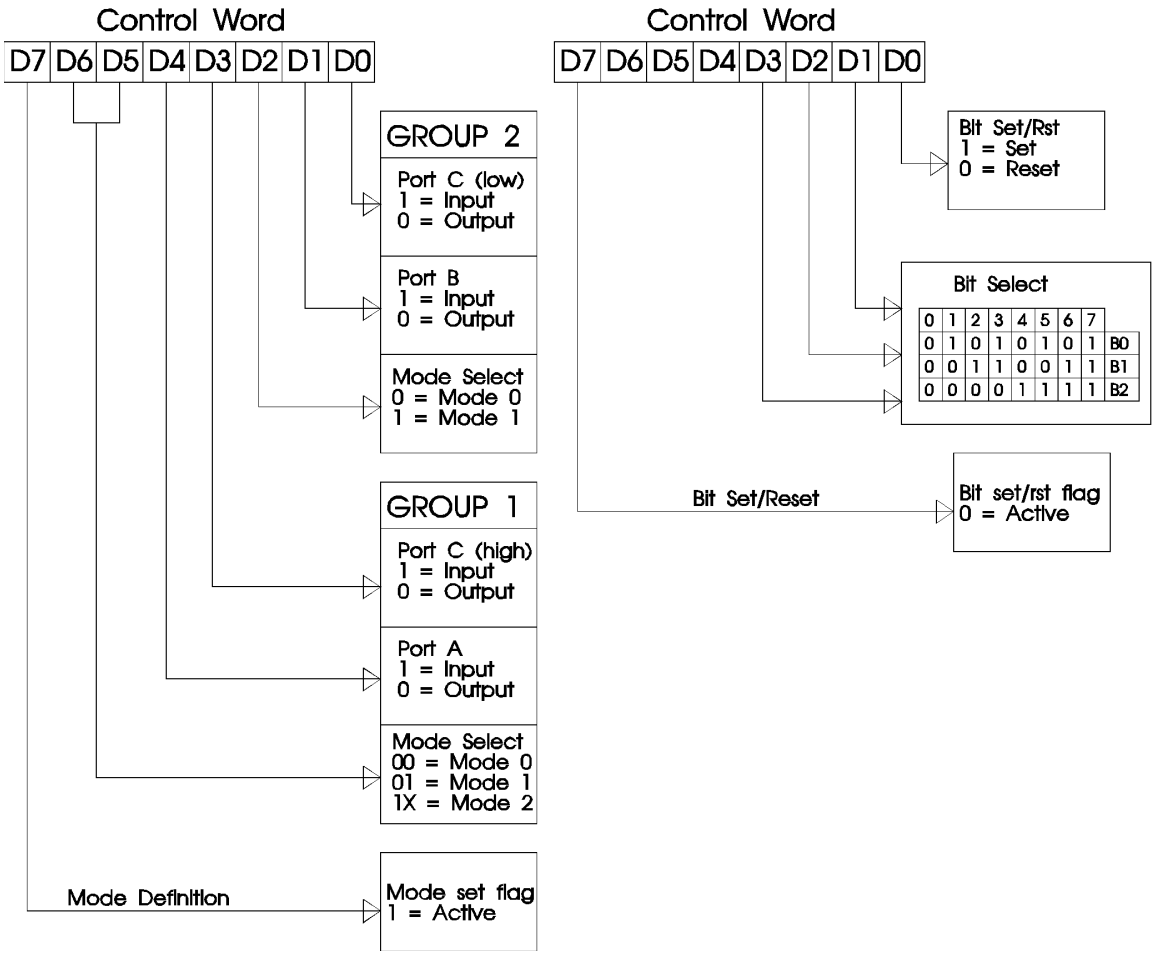
Mode 2: Strobed bi-directional input/output. Lets you communicate with an external device through port A. Handshaking is similar to mode 1.

Available Port direction definitions:

Port A may be Inputs or Outputs
Port C lower bits may be Inputs or Outputs
Port C higher bits may be Inputs or Outputs

All these modes are discussed in detail in the 8255 datasheet from Intel.

8255 Mode Definition, Bit Set/Reset Feature



Isolated optocoupler inputs

The Isolated input stage of the DM6854HR consists of 4 major parts:

1. Output data register
2. Input optocouplers
3. Input bit mask stage
4. Input bit inverter stage.

1. Output data register

The input circuitry of the DM6854HR consists of a range selection jumper field and an input protection schottky diode. The optocoupler is current driven, so the input range is selected by connecting different current limiting resistors in series with the optocoupler input diode. To protect the optocoupler input diode from overcurrent and reverse voltage, a diode is connected in reverse with the optocoupler input diode. Make sure you do not exceed the absolute maximum voltage of each nominal input range. Exceeding this limit may cause permanent damage to the optocouplers.

2. Optocouplers

The SMD-optocouplers provide the galvanic isolation between the computer and the process signals as well as isolation between input channels. The optocouplers are logically non inverting, i.e.. a voltage on the isolated digital input will be interpreted as a logical "1". To avoid false triggering on the inputs special "Schmidt trigger" buffers are used to provide reliable and precise switching of the inputs.

3. Input bit mask stage

The programmable input processing section of the DM6854HR includes an input bit mask filter stage. This mask may be is used to select bits or groups of interesting bits for pattern recognition. Bit selection is performed by writing to the mask register (BA+1). A bit is selected for comparison if it is set to one "1". The currently active mask can be read back from the same address (BA+1). After reset condition all bits are "0" and disabled.

4. Input bit inverter stage

The trigger logic of the DM6854HR uses OR-logic from the inputs. This means, that a transition from low to high on any input will cause a trigger condition. To enable bitwise selection of the polarity of the signals an inverter stage may be used. Bit inversion is performed by writing to the input inversion register (BA+2). An input channel is inverted by writing "1" to the corresponding inversion register bit. Example: outp(BA+2,0x0f); will invert input channels 1-4.

Isolated optocoupler outputs

The Isolated optocoupler output stage of the DM6854HR consists of 3 major parts:

1. Output data register
2. Schmidt triggers
3. Optocouplers.

1. Output data register

The four bit output register holds the output pattern for the output stage. Data can be updated and read from address (BA+3), bits 4-8. The lower bits (1-4) hold the configuration bits of your DM6854HR. Make sure these bits are unchanged when writing to this register. To avoid accidental changes in configuration read first data in the register (BA+3) before writing to it.

2. Schmidt triggers

The output of the optocoupler has a Schmidt trigger to condition the output of the optocouplers, this will reduce false triggering.

3. Optocouplers

Small SMD-optocouplers are used to isolate each channel of the isolated outputs. Individual optocouplers are used for each channel. The optocouplers are directly connected to the Output data register. The optocouplers are connected in a non inverting configuration in open collector configuration. The maximum switching current is 8mA at 30V.

Trigger output

Your DM6854HR is capable of causing a trigger signal caused by an input condition. This trigger output can be programmed to cause a hardware interrupt request to the host computer. This feature is especially useful when monitoring for a predefined input condition. When this predefined condition occurs, an interrupt will indicate this condition to the host. This trigger is also available as an external output signal on the 50-pin expansion connector Pin #2. You may use this to trigger other devices or other boards in your system such as the isolated digital output board and DM6854HR. The polarity of this trigger signal is programmable and is driven by a tristate buffer. This enables several boards to drive the same trigger output signal.

Interrupts

The DM6854HR has an interrupt available to indicate an external trigger. This interrupt can be used to tell the host computer that the pre-programmed input pattern condition has occurred. Chapter 4 will provide programming information on interrupts.

CHAPTER 4 - BOARD OPERATION AND PROGRAMMING

This chapter shows you how to program and use your DM6854HR. It provides a complete detailed description of the I/O-map and a detailed discussion of programming operations to aid you in application programming.

Defining the I/O Map

The I/O map of the DM6854HR is shown in Table 4-1 below. As shown the module occupies 8 addresses. The Base Address (designated as BA) can be set using the jumpers as described in Chapter 1, *Module Settings*. The following sections describe the register contents of each address used in the I/O map.

Table 4-1 DM6854HR I/O Map

Table 4-1 DM6854HR I/O Map			
Register Description	**Read Function**	**Write Function**	**Address in HEX**
Isolated inputs	Digital Inputs 1-8	Clear IRQ (trigger)	BA+0
Input Mask	Read input mask	Write to mask register	BA+1
Input Inversion	Read input inv. reg.	Write to input inv. reg.	BA+2
Isolated outputs	Read conf/output reg.	Write conf/output reg	BA+3
PPI Port A	Reads Port A data	Writes Port A data	BA+4
PPI Port B	Reads Port B data	Writes Port B data	BA+5
PPI Port C	Reads Port C data	Writes Port C data	BA+6
PPI Control Byte	Reserved	Writes to Control byte	**BA+7**
*** BA = Base Address**			

BA+0 Digital input register (Write)

The trigger condition is latched on the DM5854/6854. The interrupt request will remain high until the host acknowledges the interrupt. This is done with a write to **BA+0**.

Example: outp(BA,data); // Clears trigger and interrupt; data is irrelevant.

BA+0 Digital inputs (Read)

The eight isolated digital input bits available in connector J4 can be read from address BA+0.

BA+1 Input mask register **(Write)** 0 after reset

The input mask register selects which bits are monitored for a trigger condition. A "1" corresponds to a selected bit. This register in located in address **BA+1**.

Example: outp(BA+1, 0x7F); // Bits 1-7 are selected

BA+1 Input mask register **(Read)**

Performing a read to the input mask register address will return the current data stored in the register.

BA+2 Input inversion register **(Write)** 0 after reset

The input inversion register may be used to invert input channels selected by the input mask register. If a logical 0 on the input bit is should cause a trigger; the respective inversion bit should be set (to "1).

Example: Bits 1-7 are monitored for a condition where any bit changes state from condition 0x3E.

Step 1:	Mask required bits	outp(BA+1,0x7F);	// 0 1 1 1 1 1 1 1
Step 2:	Set inversion register	outp(BA+2,0x3E);	// 0 0 1 0 1 1 1 0

BA+2 Input inversion register **(Read)**

Performing a read to the input inversion register address will return the current data stored in the register.

BA+3 Configuration / digital output register **(Write)** 0 after reset

- BIT 0 1->External host IRQ enabled
  0->External host IRQ disabled
- BIT 1 1->External trigger output (50-pin header, pin 2) tristate buffer enabled
  0->External trigger output tristate buffer in tristate (externally pulled low)
- BIT 2 1->External trigger output active low
  0->External trigger output active high
- BITS 3-7 4-bit isolated digital output port

BA+3 Configuration / digital output register **(Read)**

Performing a read to this register address will return the current data stored in the register.

BA+4 PPI Port A (Read/Write)

Transfers the 8-bit Port A digital input and output data between the module and an external device. A read transfers data from the external device, through connector J5, and into port A; a write transfers the written data from port A through J5 to the external devices.

BA+5 PPI Port B (Read/Write)

Transfers the 8-bit Port B digital input and output data between the module and an external device. A read transfers data from the external device, through connector J5, and into port B; a write transfers the written data from port B through J5 to the external devices.

BA+6 PPI Port C (Read/Write)

Transfers the 4-bit low nibble of the Port C digital input and output data between the module and an external device. A read transfers data from the external device, through connector J5, and into bits 0-7 of port C; a write transfers the written data bits 0-7 from port C through J5 to the external devices.

BA+7 PPI Control Byte (Write only)

When bit 7 is set to 1, a write programs the PPI configuration.

D7	1 = Active	Mode Set Flag
D6	**D5**	Mode select for group #1
0	0	Mode 0
0	1	Mode 1
1	1	Mode 2

Group #1

D4	0 = A Output 1 = A Input	Direction of Port A
D3	0 = C upper Output 1 = C upper Input	Direction of Port C Upper (bits 4-7)
D2		Mode select for group #2

Group #2

D1 0 = B Output Direction of Port B
1 = B Input

D0 0 = C lower Output Direction of Port C Lower (bits 0-3)
1 = C lower Input

When bit 7 is set to 0, a write can program individual lines of port C.

D7 0 = Active Set Reset Function bit

D3	**D2**	**D1**	Bit Select
0	**0**	**0**	**PC0**
0	**0**	**1**	**PC1**
0	**1**	**0**	**PC2**
0	**1**	**1**	**PC3**
1	**0**	**0**	**PC4**
1	**0**	**1**	**PC5**
1	**1**	**0**	**PC6**
1	**1**	**1**	**PC7**

D0 1 = Set bit to 1
0 = Set bit to 0

Programming the DM6854HR

This section gives you some general information about programming the DM6854HR board, and then walks you through the major programming functions of the board. These descriptions will help you as you use the example programs and function libraries included with this board. All of the program descriptions use decimal values unless otherwise specified.

The DM6854HR is programmed by writing to and reading from the correct I/O-port addresses of the board. These I/O ports were described in the previous section of this chapter. The following example shows how to perform a 8-bit read and write I/O port addresses using "C"-syntax and assembly code:

	Read:	**Write:**
"C"-syntax	var = **inp**(address);	**outp**(address,data);
Assembly	**mov** dx,address **in** ax,dx	**mov** dx,address **mov** ax,data **out** dx,ax

Clearing and setting bits in an I/O port

When you clear or set bits in an I/O port you must be careful not to alter the status of other bits. You can preserve the status of all the bits you do not wish to change by proper use of the bitwise AND- and OR-operators. Using and or operators, single or multiple bits can easily be set or cleared in one line operations.

1) To clear a single bit in a port, AND the current value of the port with the value "B" , where B = 255-2(exp) bit.

2) To set a single bit in a port, OR the current value of the port with the value "B", where B = 2(exp) bit.

   Bits are numbered from 0-7 for the low byte of a word and from 8-15 for the high byte of a word. Setting and clearing of multiple bits in a bye or word is more complex.

3) To clear multiple bits in a port, AND the current value of the port with the value "B", where B = 255-(the sum of the values of the bits to be cleared). Note that the bits do not have to be consecutive.

4) To set multiple bits in a port, OR the current value of the port with the value "B", where B = (sum of the individual bits to be set).

Initializing the 8255 PPI

Before you can operate the non isolated general purpose digital I/O through the 8255 PPI it must be initialized. This step must be performed every time you start up, reset or reboot your computer.

The 8255 PPI is initialized by writing the appropriate control byte to the I/O Port **BA+7** The contents of your control word will vary, depending on how you want to configure your I/O lines. Use the control word description in the previous I/O map section to help you program the right value.

Example: Port A -> Output
Port B -> Output
Port C Upper -> Input
Port C Lower -> Output

Control Byte : 88h

Isolated digital input programming

The optoisolated digital inputs may be addressed directly by reading the digital input register in address (BA+0). The DM6854HR features more advanced modes of input pattern or bit change recognition to reduce software overhead. This is possible with the hardware interrupt generated by the onboard comparison logic.

The following two subsections will illustrate how to program the different modes of the isolated digital input port.

1. *Software controlled direct read*

```
in_data=inp(BA);      // Loads data from inputs into variable in_data
```

2. *Interrupt driven read upon change in bits*

```
mask = 0x55;          // Odd bits are of interest
state = 0x01;         // Any of bits 3,5,7 goes high or bit 1 goes low
outp(BA+1,mask)       // Program mask register
outp(BA+2,state);     // Program bit inversion register
outp(BA+3,0x01);      // Enable interrupts
outp(BA,0x00);        // Clear hardware interrupt , if state had already occurred

// Now if any of bits 3,5,7 goes high or bit 1 goes low it will cause a hardware
// interrupt. In your ISR. you will read the input data with " in_data = inp(BA);".
```

Isolated digital output programming

The four bit digital output port can directly be programmed from software by writing output data to the high bits (4-8) of I/O-port address BA+3. Care must be taken to ensure the lower four bits remain unchanged since they hold the configuration data of your DM6854HR. The example below illustrates a safe way of controlling the digital outputs.

Example in "C" syntax:

```
// Step 1        store the current contents of the Control/Output register
reg_data = inp(BA+3);                  // reg_data holds all information
conf_data = 0x0f && reg_data;          // mask lower bits to store configuration

// Step 2        prepare output data stored in variable out_data
out_data =  out_data<<4;               // shift left 4 bits

// Step 3        combine configuration and output data and write to port
outp(BA+3,out_data || conf_data);      // write to correct address
```

This code can be compressed into more elegant and optimum code. The steps above illustrate the logic of operation.

Trigger output programming

As described in the previous sections an external trigger may be used to indicate to the host or other devices that a predetermined input condition has occurred. Before you can use an external trigger you must first program your DM6854HR to enable correct operation. The following steps must be taken.

1. Program the mask and inversion register first here ....

2. Now we must program the polarity of the trigger output to other devices if this function is used

3. Clear interrupt and trigger register

4. Enable trigger output tristate buffer

5. Enable interrupts, this also enables the operation of the external trigger output

Interrupts

- What is an interrupt ?

An interrupt is an event that causes the processor in your computer to temporarily halt its current process and execute another routine. Upon completion of the new routine, control is returned to the original routine at the point where its execution was interrupted.

Interrupts are a very flexible way of dealing with asynchronous events. Keyboard activity is a good example; your computer cannot predict when you might press a key and it would be a waste of processor time to do nothing while waiting for a keystroke to occur. Thus the interrupt scheme is used and the processor proceeds with other tasks. Then when a keystroke occurs , the keyboard 'interrupts' the processor , and the processor gets the keyboard data , placed it into memory, and then returns to what it was doing before the interrupt occurred. Other common devices that use interrupts are network boards, A/D boards, serial ports etc.

Your DM6854HR can interrupt the main processor when an external trigger event occurs if interrupts are enabled on the board. By using interrupts you can write powerful code to deal with real world events.

-Interrupt request lines

To allow different peripheral devices to generate interrupts on the same computer, the PC XT bus has eight interrupt request channels (IRQ's). A rising edge transition on one of these lines will be latched into the interrupt controller. The interrupt controller checks to see if the interrupts are to be acknowledged from that IRQ and, if another interrupt is being processed, it decides if the new request should supersede the one in progress or if it has to wait until the one in progress is done. The priority level of the interrupt is determined by the number of the IRQ; IRQ0 has the highest priority IRQ7 the lowest. Many of the IRQ's are used by the standard system resources. IRQ0 is dedicated for the internal timer, IRQ1 is dedicated to the keyboard input, IRQ3 for serial port COM2 and IRQ4 for serial port COM1. Often interrupts 5 and 7 are free for the user.

- 8259 Programmable Interrupt Controller

The chip responsible for handling interrupt requests in a PC is the 8259 Interrupt Controller. To use interrupts you will need to know how to read and set the 8259's internal interrupt mask register (IMR) and how to send the end-of-interrupt (EOI) command to acknowledge the 8259 interrupt controller.

-Interrupt Mask Register (IMR)

Each bit in the interrupt mask register (IMR) contains the mask status of the interrupt line. If a bit is set (equal to 1), then the corresponding IRQ is masked, and it will not generate an interrupt. If a bit is cleared (equal to 0), then the corresponding IRQ is not masked, and it can generate an interrupt. The interrupt mask register is programmed through **port 21h**.

-End-of-Interrupt (EOI) Command

After an interrupt service routine is complete, the 8259 Interrupt Controller must be acknowledged by **writing the value 20h to port 20h**.

-What exactly happens when an interrupt occurs?

Understanding the sequence of events when an interrupt is triggered is necessary to correctly write interrupt handlers. When an interrupt request line is driven high by a peripheral device (such as the DM5854), the interrupt controller checks to see if interrupts are enabled for that IRQ, and then checks to see if other interrupts are active or requested and determines which interrupt has priority. The interrupt controller then interrupts the processor. The current code segment (CS), instruction pointer (IP), and flags are pushed onto the system stack, and a new set if CS and IP are loaded from the lowest 1024 bytes of memory.

This table is referred to as the interrupt vector table and each entry to this table is called an interrupt vector. Once the new CS and IP are loaded from the interrupt vector table , the processor starts to execute code from the new Code Segment (CS) and from the new Instruction Pointer (IP). When the interrupt routine is completed the old CS and IP are popped from the system stack and the program execution continues from the point it was interrupted.

-Using Interrupt in your Program

Adding interrupt support to your program is not as difficult as it may seem especially when programming under DOS. The following discussion will cover programming under DOS. Note, that even the smallest mistake in your interrupt program may cause the computer to hang up and will only restart after a reboot. This can be frustrating and time-consuming.

-Writing an Interrupt Service Routine (ISR)

The first step in adding interrupts to your software is to write an interrupt service routine (ISR). This is the routine that will be executed automatically each time an interrupt request occurs for the specified IRQ. An ISR is different from other subroutines or procedures. First, on entrance the processor registers must be pushed onto the stack before anything else! Second, just before exiting the routine , you must clear the interrupt on the DM6854HR by writing to address BA+0, and write the EOI command to the interrupt controller. Finally, when exiting the interrupt routine the processor registers must be popped from the system stack and you must execute the IRET assembly instruction. This instruction pops the CS, IP and processor flags from the system stack. These were pushed onto the stack when entering the ISR.

Most compilers allow you to identify a function as an interrupt type and will automatically add these instructions to your ISR with one exception: most compilers do not automatically add the EOI command to the function, you must do it yourself. Other than this and a few exceptions discussed below, you can write your ISR as any code routine. It can call other functions and

procedures in your program and it can access global data. If you are writing your first ISR, we recommend you stick to the basics; just something that enables you to verify you have entered the ISR and executed it successfully. For example: set a flag in your ISR and in your main program check for the flag.

Note: If you choose to write your ISR in in-line Assembly , you must push and pop registers correctly, and exit the routine with the IRET instruction instead of the RET instruction.

There are a few precautions you must consider when writing ISR's. The most important is, **do not use any DOS functions or functions that call DOS functions from an interrupt routine.** DOS is not reentrant; that is, a DOS function cannot call itself. In typical programming, this will not happen because of the way DOS is written. But what about using interrupts? Then, you could have the situation such as this in your program. If DOS function X is being executed when an interrupt occurs and the interrupt routine makes a call to the DOS function X, then function X is essentially being called while active. Such cases will cause the computer to crash. DOS does not support such operation. A general rule is , that do not call any functions that use the screen, read keyboard input and any file I/O routines should not be used in ISR's.

The same problem of reentrancy exists for many floating point emulators as well, meaning you should avoid floating point mathematical operations in your ISR.

Note, that the problem of reentrancy exists, no matter what programming language you use. Even, if you are writing your ISR in Assembly language, DOS and many floating point emulators are not reentrant. Of course, there are ways to avoid this problem, such as those which involve checking if any DOS functions are currently active when your ISR is called, but such solutions are beyond the scope of this manual.

The second major concern when writing ISR's is to make them as short as possible in term of execution time. Spending long times in interrupt service routines may mean that other important interrupts are not serviced. Also, if you spend too long in your ISR, it may be called again before you have exited. This will lead to your computer hanging up and will require a reboot.

Your ISR should have the following structure:

- Push any processor registers used in your ISR. Most C compiler do this automatically
- Put the body of your routine here
- Clear the interrupt bit by writing to address BA+3
- Issue the EOI command to the 8259 by writing 20h to address 20h
- Pop all registers. Most C compilers do this automatically

The following C example shows what the shell of your ISR should be like:

```
/*---------------------------------------------------------------------------
| Function:     new_IRQ_handler
| Inputs:       Nothing
| Returns:      Nothing         - Sets the interrupt flag for the EVENT.
|---------------------------------------------------------------------------*/
void interrupt far new_IRQ_handler(void)
{

        IRQ_flag = 1;           // Indicate to main process interrupt has occurred
        {
        // Your program code should be here
        }
        outp(BA+0,0x00);    // Clears interrupt on DM6854HR
        outp(0x20, 0x20);   /* Acknowledge the interrupt controller. */
}
```

-Saving the Startup Interrupt Mask Register (IMR) and interrupt vector

The next step after writing the ISR is to save the startup state of the interrupt mask register (IMR) and the original interrupt vector you are using. The IMR is located in address **21h**. The interrupt vector you will be using is located in the interrupt vector table which is an array of 4-byte pointers (addresses) and it is locate din the first 1024 bytes of the memory (Segment 0 offset 0). You can read this value directly, but it is a better practice to use DOS function 35h (get interrupt vector) to do this. Most C compilers have a special function available for doing this. The vectors for the hardware interrupts on the XT - bus are vectors 8-15., where IRQ0 uses vector 8 and IRQ7 uses vector 15. Thus if your DM6854HR is using IRQ5 it corresponds to vector number 13.

Before you install your ISR, temporarily mask out the IRQ you will be using. This prevents the IRQ from requesting an interrupt while you are installing and initializing your ISR. To mask the IRQ, read the current IMR at I/O port 21h, and set the bit that corresponds to tout IRQ. The IMR is arranged so that bit 0 is for IRQ0 and bit 7 is for IRQ7. See the paragraph entitled *Interrupt Mask Register (IMR*) earlier in this discussion for help in determining your IRQ's bit. After setting the bit, write the new value to I/O port 21h.

with the startup IMR saved and the interrupts temporarily disabled, you can assign the interrupt vector to point to your ISR. Again you can overwrite the appropriate entry in the vector table with a direct memory write, but this is not recommended. Instead use the DOS function 25h (Set Interrupt Vector) or, if your compiler provides it, the library routine for setting up interrupt vectors. Remember , that interrupt vector 8 corresponds to IRQ0 , vector 9 for IRQ1 etc.

If you need to program the source of your interrupts, do that next. For example, if you are using the input compare on the DM6854HR as an interrupt source, program it to do that. Finally, clear the mask bit for your IRQ in the IMR. This will enable your IRQ.

-Common Interrupt mistakes

- Remember, hardware interrupts are from 8-15, IRQ's are numbered 0-7
- Forgetting to clear the IRQ mask bit in the IMR
- Forgetting to send the EOI command after ISR code. Disables further interrupts.

Example on Interrupt vector table setup in C-code:

```
void far _interrupt new_IRQ1_handler(void );        /* ISR function prototype */
#define IRQ1_VECTOR       3                         /* Name for IRQ */
void (interrupt far *old_IRQ1_dispatcher)
        (es,ds,di,si,bp,sp,bx,dx,cx,ax,ip,cs,flags);      /* Variable to store old IRQ_Vector */
void far _interrupt new_IRQ1_handler(void );

/*---------------------------------------------------------------------
| Function:     init_irq_handlers
| Inputs:       Nothing
| Returns:      Nothing
| Purpose:      Set the pointers in the interrupt table to point to
|               our funtions ie. setup for ISR's.
|---------------------------------------------------------------------*/
void init_irq_handlers(void)
{
        _disable();
                old_IRQ1_handler = _dos_getvect(IRQ1_VECTOR + 8);
                _dos_setvect(IRQ1_VECTOR + 8, new_IRQ1_handler);
                Gi_old_mask = inp(0x21);
                outp(0x21,Gi_old_mask & ~(1 << IRQ1_VECTOR));
        _enable();
}
```

```
|/*---------------------------------------------------------------------
| Function:     restore do this before exiting program
| Inputs:       Nothing
| Returns:      Nothing
| Purpose:      Restore interrupt vector table.
|---------------------------------------------------------------------*/
void restore(void)
{
        /* Restore the old vectors */

        _disable();

                _dos_setvect(IRQ1_VECTOR + 8, old_IRQ1_handler);
                outp(0x21,Gi_old_mask);

        _enable();

}
```

APPENDIX A

==========

DM5854HR/DM6854HR Specifications

Interface

Jumper-selectable base address, I/O mapped
Jumper selectable interrupts

Digital I/O (non isolated)

Number of lines	**24**
Logic compatibility	**TTL/CMOS**

(Pullup and Pulldown resistors are available)

Digital inputs (isolated)

Number of lines	**8**
Input ranges (jumper selectable)	**4**
Triggering Voltages +5V range	**3,3V** (approx)
+12V range	**8,5V** (approx)
+24V range	**20V** (approx)
Isolation Voltage	**1.500V Rms**
Optocoupler switching current	**10mA (Typical)**
	16mA (Maximum)

Digital outputs (isolated)

Number of lines	**4**
Isolation Voltage	**1.500V Rms**
Output stage	**Open collector, 8mA@30V**

Connectors

Isolated I/O	**24 terminal screw terminal connector**
	Optional header connector 48-Pin
Non Isolated I/O	**Header connector 50-Pin**

NOTES: